Converting Handwritten Mathematical Expressions into LaTeX

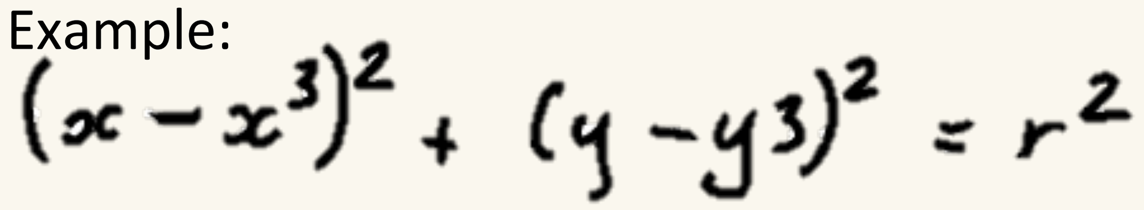
Norah Borus (nborus), William Bakst (wbakst), Amit Schechter (amitsch)

Motivation

- Typing up mathematical equations has become quite a necessity when submitting academic papers, writing and solving problem sets, presenting mathematical concepts, and much more.
- We're looking to automate the process of converting handwritten expressions to LaTeX using modern Machine Learning techniques.

Datasets

- Our dataset includes the traces of 10,000 handwritten expressions mapped to the corresponding ground truth LaTeX expression (obtained from Kaggle).
- We split the data into 80% train, 10% dev, and 10% test. We use traces for **CSeg**, normalized pixel arrays for **OCR**, and traces with segmented characters for **SA**.
- Example:



\$(x-x^3)^2 + (y-y^3)^2 = r^2\$

Features

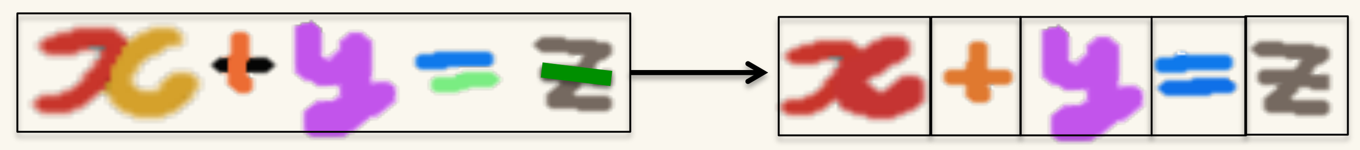
- <u>Segmentation:</u> features come from data. SVM uses features including overlap, inverse horizontal, vertical, and centroids distance, and horizontal containment, which are the main indicators for grouping traces.
- <u>Character Recognition:</u> SVM and NN use data based features include normalized flattened greyscale pixel array.

We derived Histogram of Oriented Gradients (HOG) from the pixel array – improves image recognition by using overlapping local contrast normalization.

Methods

Character Segmentation (CSeg)

<u>**Input:**</u> list of all traces for equation. <u>**Output:**</u> traces grouped by characters.



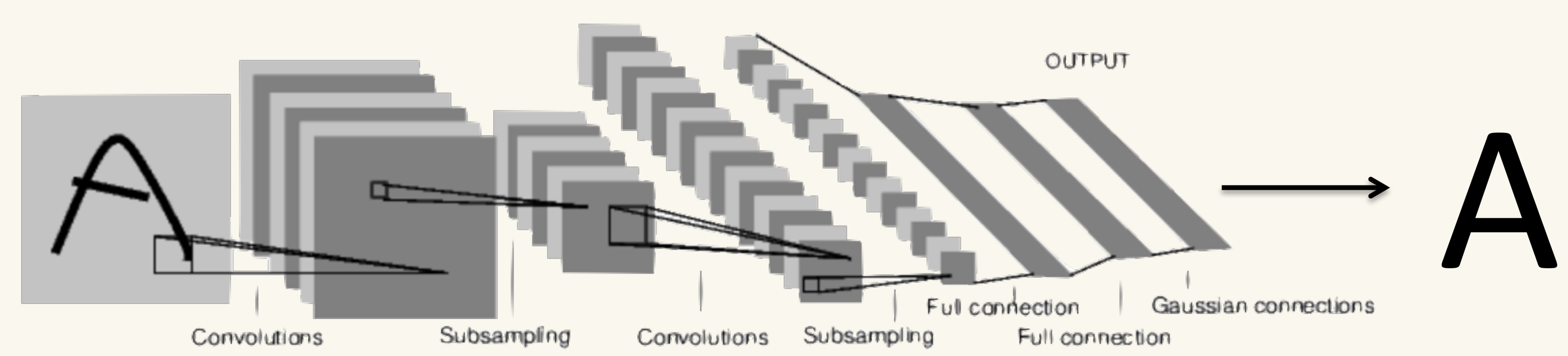
- <u>**Overlap:**</u> merges overlapping strokes. All other strokes are separate.
- <u>**SVM:**</u> uses a binary SVM with a linear Kernel, L_2 regularization, and C value 50. Binary SVM loss function: $L_i = C\max(0, 1 - y_i w^T x_i) + R(W)$
- <u>**NN (binary classification):**</u> with single hidden layer, softmax and sigmoid activation, and cross entropy loss. Input is a 32X32 flattened pixel array of a single or multiple strokes. Output is whether or not the image is a valid LaTex symbol.
- <u>**Beam Search:**</u> we use beam search to explore possible segmentations. We maximize the segmentation score, which is based on NN prediction (confidence of it being a valid character).

$$score = \underset{seg \in S}{\operatorname{argmax}} \frac{\sum_{g \in seg} P(g)}{|seg|}$$

Character Recognition (OCR)

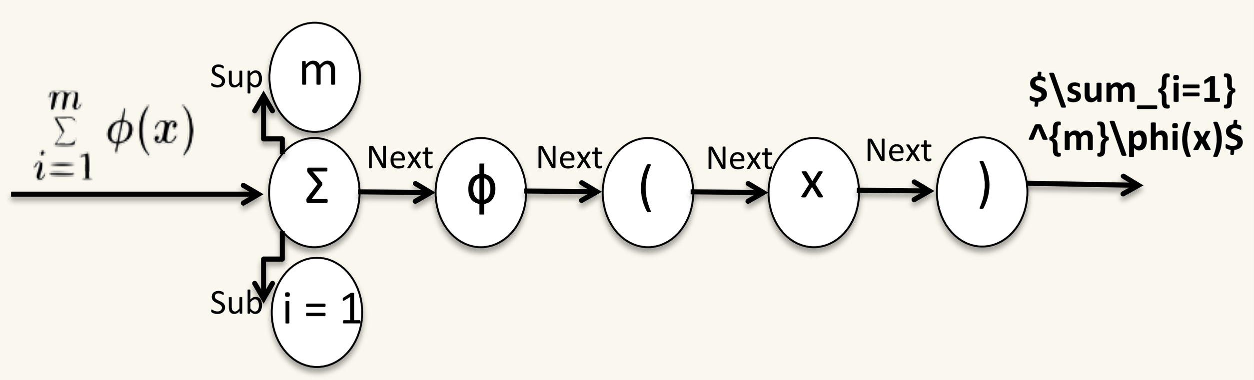
<u>**Input:**</u> normalized pixel array. <u>**Output:**</u> character's LaTex representation.

- <u>**SVM:**</u> uses multiclass SVM with a linear Kernel, L_2 regularization. Multiclass SVM loss: $L = \frac{1}{N}\sum_i \sum_{j \neq y_i} [\max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \Delta)] + \lambda \sum_k \sum_l W_{k,l}^2$
- <u>**NN (multiclass):**</u> uses same internal structure as NN in CSeg. Output is the classification of the LaTex symbol.
- Cross entropy loss: $CE(y, \hat{y}) = -\sum_{k=1}^{K} y_k \log \hat{y_k}$
- <u>**CNN:**</u> multiclass CNN with 5 hidden layers, ReLU activation function, and cross entropy loss.



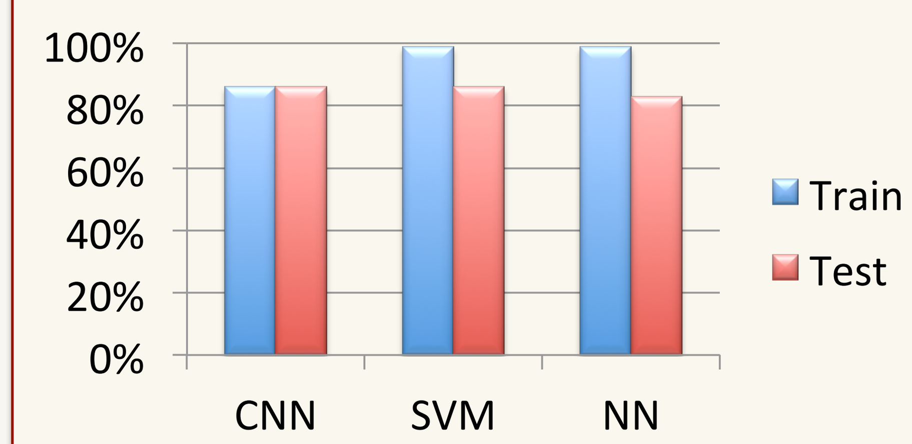
Structural Analysis (SA)

- Structure analysis is done according to relative location and size.
- We use features such as overlap, squared loss from superscript/subscript bounding box to determine the relationship between two characters.
- We then use these relationships to rebuild the ground truth LaTeX.



Experimental Results

<u>OCR Accuracy:</u>



<u>SA Analysis Accuracy:</u>

	Baseline	Heuristics
Test	14%	65%

<u>CSeg Accuracy:</u>

	Test	Train
CNN	44%	NA
Overlap	53%	53%
SVM	64%	96%
Binary NN	78%	97%
Beam search	64%	NA

Discussion

- Most character recognition errors come from symbols that do not appear as frequently in dataset (e.g. \mu).
- Segmentation: NN beam search and the binary SVM achieve similar accuracy (64%). Beam search tends to group traces that should remain separated. Binary SVM tends to keep strokes separated.
- We are working on improving our Structural Analysis. We are struggling with how to accurately test the output and train models and are currently reporting accuracy by hand to preserve correctness .

Future Work

- <u>**CSeg:**</u> implement a combined model using NN beam search and SVM predictions to increase accuracy. Use AdaBoost and multi-scale shape context features.
- <u>**OCR:**</u> Experiment with advanced OCR techniques such as Random Forests to help improve the accuracy of our CNN and SVM. Use data augmentation techniques to add examples of uncommon symbols.
- <u>**SA:**</u> We plan on finding or implementing a LaTeX parser that will enable us to train an SVM to aid our current model in correctly building square root, exponents, and other more complex equations.
- <u>**E2E:**</u> implement a CNN for an end to end approach (similar to Google's Tesseract).

<u>References:</u>

- M. Thoma – "on-line recognition of Handwritten Mathematical Symbols", 2015, *Bachelor'sTthesis, KIT.*
- Scikit Learn. http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html.
- Pytorch. http://pytorch.org/tutorials/beginner/blitz/neural_networks_tutorial.html.